迁移强化学习综述(part 1)

田鸿龙

Transfer Learning in Deep Reinforcement Learning: A Survey

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou

Transfer Learning Approaches

- **Reward Shaping(today)**
- Learning from Demonstrations
- Policy Transfer
- Inter-Task Mapping
- Reusing Representations
- Learning Disentangled Representations

Reward Shaping

- The agent will learn its policy using the newly shaped rewards R' : $R' = R + F$
- $F : S \times S \times A \rightarrow \mathbb{R}$
- $M = (S, A, T, \gamma, R)) \rightarrow M' = (S, A, T, \gamma, R')$
- Reward Shaping **leverages prior knowledge** to reconstruct the reward distributions for the target MDP to bias the agent's action selections

Cycle Dilemma

- 如果agent绕着轨迹 $(s_1 \to s_2 \to \cdots \to s_n \to s_1 \to \cdots)$不停的走下去，同时它们对应的 F满足

$$F(s_1, a_1, s_2) + \cdots + F(s_{n-1}, a_{n-1}, s_n) + F(s_n, a_n, s_1) > 0$$

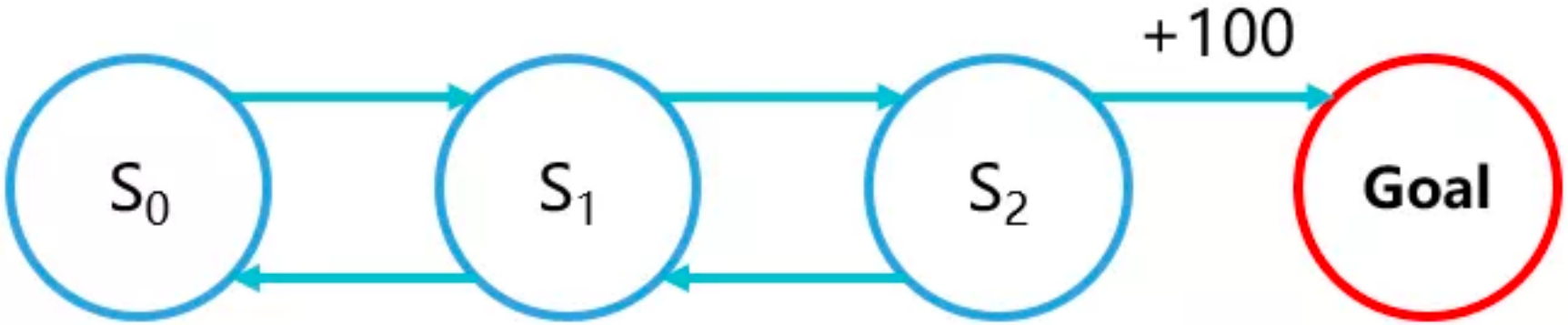
- 那么这个agent每绕一圈就会增加一些奖赏，即便它们原本的奖赏可能是0，也就是

$$R(s_n, a_n, s_1) = R(s_1, a_1, s_2) = \cdots = R(s_{n-1}, a_{n-1}, s_n) = 0$$

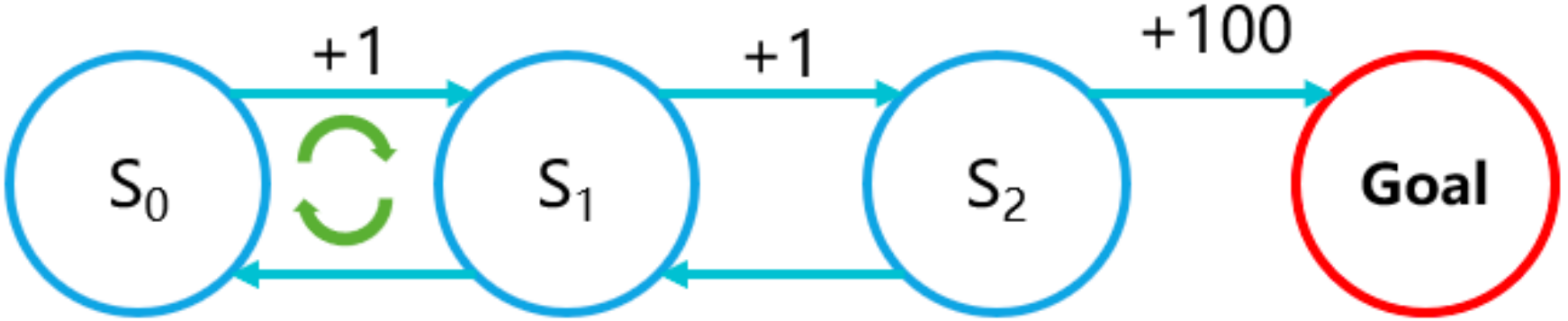
- 我们称这时agent “distracted” 了

Cycle Dilemma(cont.)

- MDP without reward shaping:



- MDP with bad reward shaping:



Potential based Reward Shaping

- Potential based Reward Shaping (PBRS) is the most classical RS approach.

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s)$$

- PBRS的思想： Policy invariance，即在提升训练速度的同时保证得到的策略和通过原来的reward的策略一致。
- 可以直接验证Cycle Dilemma的问题已经被解决：

$$F(s_1, a_1, s_2) + \cdots + F(s_{n-1}, a_{n-1}, s_n) + F(s_n, a_n, s_1) = 0$$

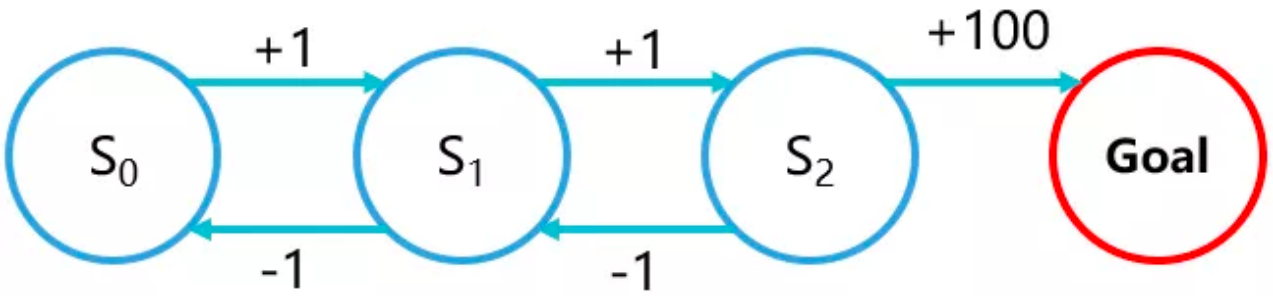
- without further restrictions on the underlying MDP or the shaping function F, PBRS is **sufficient** and **necessary** to preserve the policy invariance.

$$Q^*_{M'}(s, a) = Q^*_M(s, a) - \Phi(s)$$

A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in ICML, vol. 99, 1999, pp. 278–287.

Potential based Reward Shaping

- MDP with Potential based Reward Shaping



$\Phi(s_0) = 0$ $\Phi(s_1) = 1$ $\Phi(s_2) = 2$

Proof of sufficient

原Bellman方程两边同时减去一个$\Phi(s)$

$$Q_M^*(s,a) - \Phi(s)$$
$$= \mathbb{E}_{s' \sim P_{sa}}[R(s,a,s') + \gamma \max_{a' \in A} Q_M^*(s',a')] - \Phi(s)$$
$$= \mathbb{E}_{s' \sim P_{sa}}\left[R(s,a,s') + \gamma\Phi(s') + \gamma \max_{a' \in A}\left(Q_M^*(s',a') - \Phi(s')\right)\right] - \Phi(s)$$
$$= \mathbb{E}_{s' \sim P_{sa}}\left[R(s,a,s') + \gamma\Phi(s') - \Phi(s) + \gamma \max_{a' \in A}\left(Q_M^*(s',a') - \Phi(s')\right)\right]$$

$$\pi_{M'}^*(s) \in \arg\max_{a \in A} Q_{M'}^*(s,a)$$
$$= \arg\max_{a \in A} Q_M^*(s,a) - \Phi(s)$$
$$= \arg\max_{a \in A} Q_M^*(s,a)$$

Potential Based state-action Advice

- PBA是在PBRS的基础之上的扩展

$$F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a)$$

- 同样满足

$$Q^*_{M'}(s,a) = Q^*_M(s,a) - \Phi(s,a)$$

- 这意味着，尽管PBA学习出的policy和通过原reward得到的policy不一致，但可以通过$Q^*_{M'}(s,a)$计算出最优policy

$$\pi^{\pi}_M(s) = \underset{a\in A}{\operatorname{argmax}}\left(Q^*_{M'}(s,a) + \Phi(s,a)\right)$$

Dynamic Potential Based

- 在势函数中加入时间参量p

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t)$$

- this dynamic approach can still maintain policy invariance where t is the current time :

$$Q^*_{M'}(s,a) = Q^*_M(s,a) - \Phi(s,t)$$

- Ps: 允许势能函数随时间变化，使得训练一个势能函数成为可能。

Dynamic value-function Advice

- Intention：如果有一个任意的奖励函数，能不能把它改造成基于势能的奖励函数？

- 假设存在专家给出的函数$F: S \times A \to \mathbb{R}$ 用于评价任何状态动作对（这个函数可以是另一个agent学习出来的Q函数），是否可以直接令$\Phi(s,a) = F(s,a)$？

- 下面通过一个例子说明直接令$\Phi(s,a) = F(s,a)$不仅没有充分利用专家知识，反而起到了相反的效果。

Dynamic value-function Advice

- Example：假设专家认为某个(s,a)很好，设$\mathrm{F}(s,a)=1$，其余任意$\mathrm{F}(s,a)=0$
- 然而，有

$$F(s,a,s',a')=\Phi(s',a')-\Phi(s,a)=0-1=-1$$

- 因此，实际上对效果反而是相反的。

- 实际上，如果专家认为(s,a)很好，应该让$\Phi(s',a')$尽可能大，其中s'是agent在状态s下，采取动作a（可能）到达状态。
- 然而，这要求MDP的相关信息。

Dynamic value-function Advice

- Dynamic value-function Advice(DPBA)的目标是学出一个函数F^{Φ}，使得$F^{\Phi}(s,a) \approx R^{\dagger}(s,a)$，其中 $R^{\dagger}$为专家给出的建议。
- 其中，$F^{\Phi}(s,a) = \gamma\Phi(s',a') - \Phi(s,a)$，这意味着学习到的$F^{\Phi}(s,a)$满足之前讨论到的良好性质。
- 于是问题变成了如何学习函数$\Phi(s,a)$。
- 回忆一下Q函数的定义：

$$\mathrm{Q}(s,a) = r(s,a) + \gamma Q(s',a')$$

- 把上面的公式整理一下

$$\Phi(s,a) = -F^{\Phi}(s,a) + \gamma\Phi(s',a')$$

- 其中$F^{\Phi}(s,a) = R^{\dagger}(s,a)$

Dynamic value-function Advice

- Q-learning:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \delta_t$$
$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

- Dynamic value-function Advice:
- 令$R^{\Phi} = -R^{\dagger}$

$$\Phi_{t+1}(s, a) = \Phi_t(s, a) + \beta_t \delta_t^{\Phi}$$
$$\delta_t^{\Phi} = r_{t+1}^{\Phi} + \gamma \Phi_t(s_{t+1}, a_{t+1}) - \Phi_t(s_t, a_t)$$

- 当算法收敛时，有

$$F(s, a) = \gamma \Phi(s', a') - \Phi(s, a) = R^{\dagger}(s, a)$$

Paper

Policy Transfer using Reward Shaping

Tim Brys
Vrije Universiteit Brussel
timbrys@vub.ac.be

Anna Harutyunyan
Vrije Universiteit Brussel
aharutyu@vub.ac.be

Matthew E. Taylor
Washington State University
taylorm@eecs.wsu.edu

Ann Nowé
Vrije Universiteit Brussel
anowe@vub.ac.be

Idea

- 存在映射χ_S和χ_A满足

$$\chi_S\left(s_{\text{target}}\right) = s_{\text{source}}$$

$$\chi_A\left(a_{\text{target}}\right) = a_{\text{source}}$$

- 同时，已知在source domain中的专家策略π
- 方法一：使用DPBA的方法，令

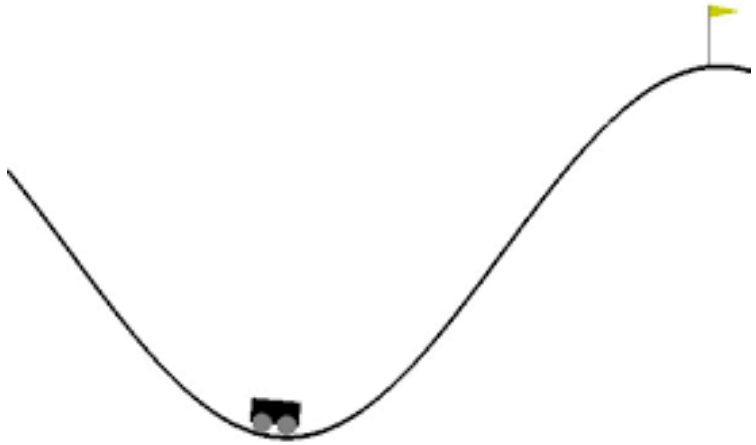
$$R^{\dagger} = \pi(\chi_S(s), \chi_A(a))$$

- 方法二：使用PBA的方法，令

$$\Phi = = \pi(\chi_S(s), \chi_A(a))$$

Experiment: Mountain Car

- Mountain Car 2D
  - the agent is in control of a car and needs to drive this car up a hill. Yet, the car is underpowered, and therefore cannot drive up the hill in one go.
  - State space: (x, v_x)
  - Action space: Left, Right, Neutral
- Mountain Car 3D
  - State space: (x, v_x, y, v_y)
  - Action space: East, West, North, South, Neutral



Experiment: Mountain Car

Mapping:

$$\chi_S((x, \dot{x}, y, \dot{y})) = \begin{cases} (x, \dot{x}) & \text{probability } 0.5 \\ (y, \dot{y}) & \text{probability } 0.5 \end{cases}$$

$$\chi_A(\text{West}) = \text{Left}$$

$$\chi_A(\text{South}) = \text{Left}$$

$$\chi_A(\text{East}) = \text{Right}$$

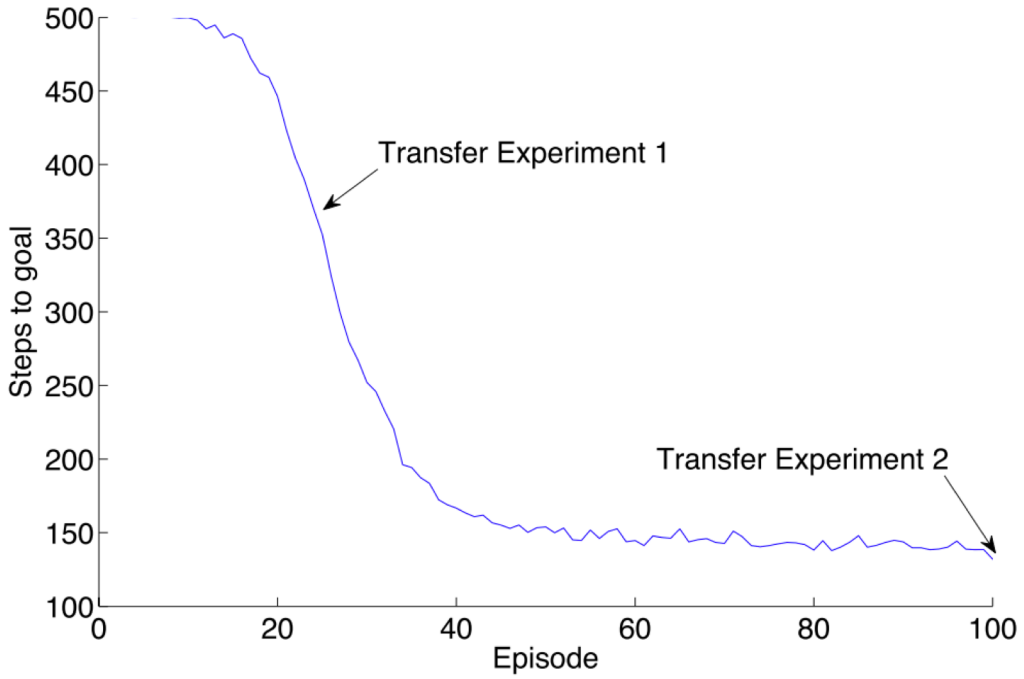
$$\chi_A(\text{North}) = \text{Right}$$

$$\chi_A(\text{Neutral}) = \text{Neutral}$$

$$R^{\pi}(s, a, s') = \frac{\pi(\chi_{S,1}(s), \chi_A(a)) + \pi(\chi_{S,2}(s), \chi_A(a))}{2}$$

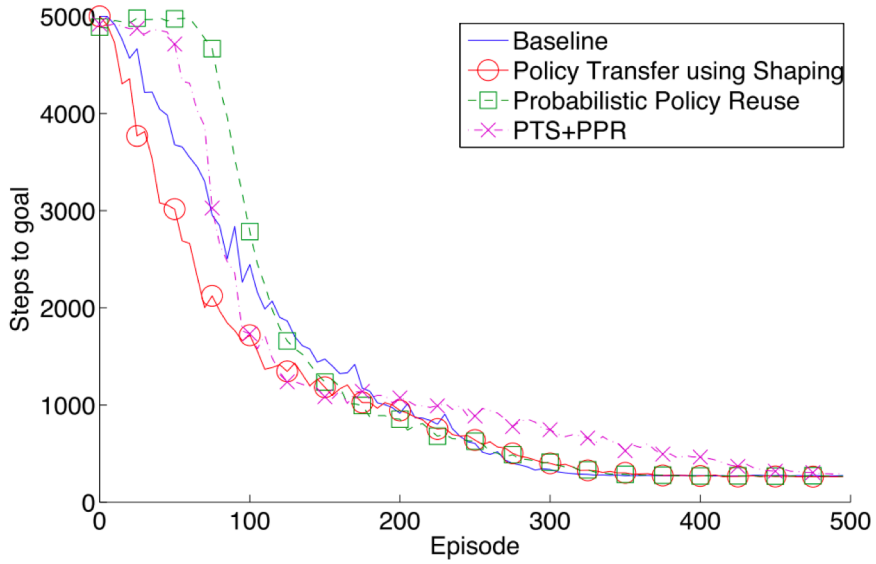
Experiment: Mountain Car

- transferring after 25 or 100 episodes of learning in the 2D Mountain Car task

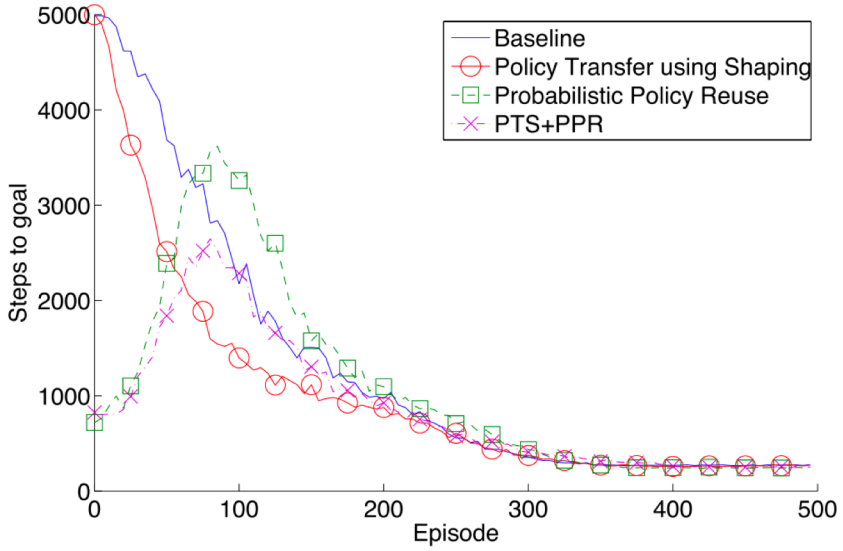


Experiment: Mountain Car

- 25 episodes



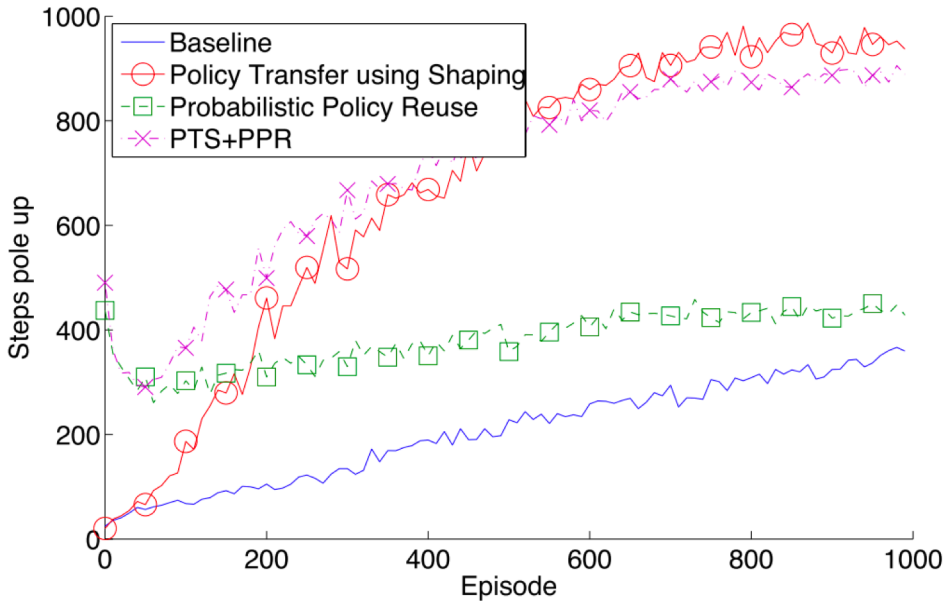
- 100 episodes



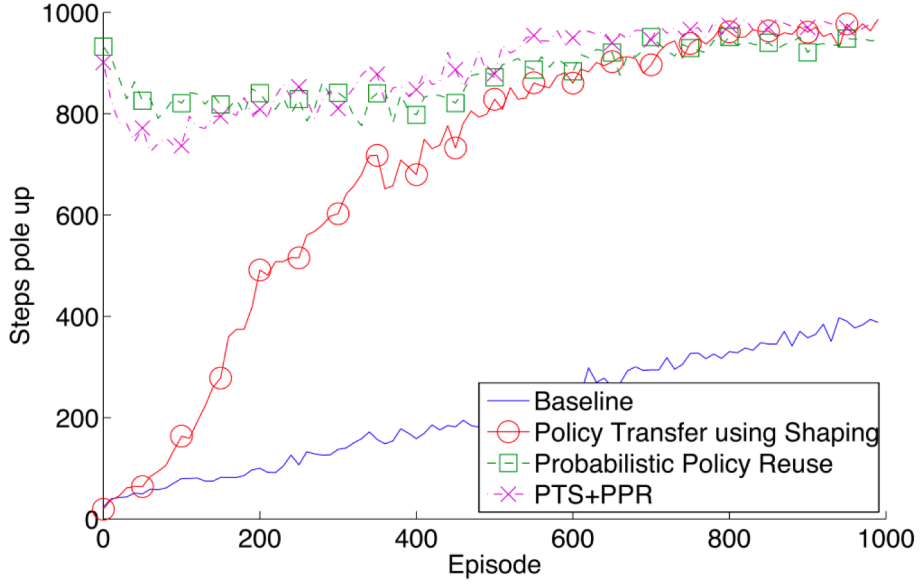
Experiment: Cart Pole

- source task: standard Cart Pole task
- target task: Heavy Cart Pole task, as we increase the weight of the pole from 0.1 to 1.0
- no state or action mappings are required for transfer

25:



100:



Experiment: Mario

- Action:
  $\{left, right, no\ direction\} \times \{jump, do\ not\ jump\} \times \{run, do\ not\ run\}$
- State:
  - Source domain: four boolean variables
  - Target domain: four boolean variables + info about enemies

- Source domain: without enemies
- Target domain: with enemies

Experiment: Mario

- Transfer when 25 and 100 episode